RESEARCH ARTICLE

Towards a General Approach for Bat Echolocation Detection and Classification

Oisin Mac Aodha[1,2] | Santiago Martínez Balvanera[3] | Elise Damstra[4] | Martyn Cooke[5] | Philip Eichinski[6] | Ella Browning[3,7] | Michel Barataud[8] | Katherine Boughey[7] | Roger Coles[9] | Giada Giacomini[7] | M. Cristina Mac Swiney G.[10] | Martin K. Obrist[11] | Stuart Parsons[12] | Thomas Sattler[13] | Kate E. Jones[3]

[1]School of Informatics, University of Edinburgh, UK

[2]The Alan Turing Institute, UK

[3]Centre for Biodiversity and Environmental Research, University College London, UK

[4]Department of Life Sciences, Imperial College London, UK

[5]Charlwood, UK

[6]School of Electrical Engineering and Computer Science, Queensland University of Technology, Australia

[7]Bat Conservation Trust, UK

[8]Combeauvert, 23250 Thauron, France

[9]Nanobat Systems, Brisbane, Australia

[10]Centro de Investigaciones Tropicales, Universidad Veracruzana, Xalapa, México

[11]Swiss Federal Institute for Forest, Snow and Landscape Research WSL, Switzerland

[12]University of the Sunshine Coast, Australia

[13]Swiss Ornithological Institute, Switzerland

Correspondence
Oisin Mac Aodha
Email: oisin.macaodha@ed.ac.uk

Funding information

Abstract

1. Acoustic monitoring is an effective and scalable way to assess the health of important bioindicators like bats in the wild. However, the large amounts of resulting noisy data requires accurate tools for automatically determining the presence of different species of interest. Machine learning-based solutions offer the potential to reliably perform this task, but can require expertise in order to train and deploy.

2. We propose BatDetect2, a novel deep learning-based pipeline for jointly detecting and classifying bat species from acoustic data. Distinct from existing deep learning-based acoustic methods, BatDetect2's outputs are interpretable as they directly indicate at what time and frequency a predicted echolocation call occurs. BatDetect2 also makes use of surrounding temporal information in order to improve its predictions, while still remaining computationally efficient at deployment time.

3. We present experiments on five challenging datasets, from four distinct geographical regions (UK, Mexico, Australia, and Brazil). BatDetect2 results in a mean average precision of 0.88 for a dataset containing 17 bat species from the UK. This is significantly better than the 0.71 obtained by a traditional call parameter extraction baseline method.

4. We show that the same pipeline, without any modifications, can be applied to acoustic data from different regions with different species compositions. The data annotation, model training, and evaluation tools proposed will enable practitioners to easily develop and deploy their own models. BatDetect2 lowers the barrier to entry preventing researchers from availing of effective deep learning bat acoustic classifiers. Open source software is provided at:

`https://github.com/macaodha/batdetect2`

KEYWORDS

Bioacoustics, Bats, Passive Acoustic Monitoring, Deep Learning, Acoustic Event Detection

1 | INTRODUCTION

Reliable biodicators are necessary to enable us to better measure the impact of climate change and accelerating habitat loss. Bats have previously been identified as one promising candidate for this role due to their global distribution, taxonomic diversity, and sensitivity to environmental and habitat change (Jones et al., 2009). However, despite making up approximately one fifth of all mammalian diversity, we know comparatively less about them in relation to other well studied taxonomic groups (Frick et al., 2020). In order for them to fulfill this potential, there is a growing need for robust and reliable tools for monitoring their populations (Russo et al., 2021).

Recent advances in hardware and software have resulted in low-cost solutions for automated bioacoustic monitoring. This enables us to unobtrusively monitor wild populations at unprecedented spatial and temporal scales via audio (Gibb et al., 2019). In the context of bats, there is a rich history of using acoustic methods for monitoring purposes (Zamora-Gutierrez et al., 2021) by leveraging the fact that bats use sound to navigate and communicate (Jones and Siemers, 2011; Prat et al., 2016). Machine learning-based approaches have been extensively used by extracting acoustic features from audio recordings and then classifying which species are present in the input audio (Parsons and Jones, 2000; Walters et al., 2012; Zamora-Gutierrez et al., 2016; Bas et al., 2017; Obrist and Boesch, 2018; Roemer et al., 2021). In this line of work, the extracted features are typically manually crafted to encode discriminative information related to the temporal and frequency-based characteristics of bat echolocation calls.

However, bat calls are complex and varied. They can exhibit regional, habitat, and species-specific variation which makes them challenging to precisely characterise using hand-crafted features (Walters et al., 2013; Russo et al., 2018). This is in addition to other complicating factors such as background noise and other vocalising species (e.g. small mammals and insects) that can be present in ultrasonic audio recordings. Deep learning-based approaches attempt to address these challenges by learning discriminative representations directly from the raw input data. They have been shown to be highly successful across a wide variety of applications in ecological monitoring (Christin et al., 2019), in addition to bioacoustics (Stowell, 2022).

The first deep learning-based methods applied to bat acoustic monitoring focused on determining the presence of bats versus background noise (Mac Aodha et al., 2018) or the species present (Chen et al., 2020; Kobayashi et al., 2021; Khalighifar et al., 2022) from short audio clips, i.e. typically shorter than 50 milliseconds. The disadvantage of these approaches is that they cannot capture longer temporal information such as the interval between individual pulses which can sometimes be an important discriminative signal. To address this issue, other work has used longer input recordings in order to capture multiple individual calls in a sequence (Paumen et al., 2021; Zualkernan et al., 2020; Tabak et al., 2021). Unfortunately the higher dimensionality of the data, due to the longer input audio recording, can necessitate larger models and thus require more supervised data at training time. Compact and efficient models are necessary in the context of low powered deployments on edge-based monitoring devices (Gallacher et al., 2021; Zualkernan et al., 2021). In addition, there is also an increased chance in longer recordings that more than one species could be present in the longer input recording (Dierckx et al., 2022). This last point is especially problematic as it violates the 'one species per input' assumption of conventional classification approaches.

Despite this recent progress in deep learning-based solutions for bat monitoring, there is still a gap between the latest research advances and the open-source tools available to practitioners. In this work, we attempt to address this gap by proposing a novel pipeline for bat echolocation call detection and species classification from acoustic data. Our approach, called BatDetect2, combines the strengths of the short temporal window-based methods with the benefits of the longer-range temporal reasoning of the call sequence based methods. Our main contributions are: (i) An efficient model for joint detection and classification of bat echolocation

calls. (ii) This model provides interpretable predictions that illustrate where in the input spectrogram, in terms of frequency and time, the model has detected a call. (iii) We evaluate the effectiveness of our proposed approach on five challenging datasets, collected from four distinct geographical regions, and show that it is superior to existing call parameter-based methods. (iv) We provide open-source tools for our full pipeline in order to enable practitioners to annotate data, train, and deploy models on their own datasets.[1]

2 | MATERIALS AND METHODS

2.1 | Acoustic event detection

Distinct acoustic vocalisation events (e.g. a bat echolocation call or a bird song) created by a species of interest can be characterised by the start time of the event, the duration of the event, and the minimum and maximum frequency bands that the event spans. Our goal is to develop a model $g()$ that takes an ultrasonic audio recording as input, represented as a spectrogram $\mathbf{x}$, and outputs a set of predictions related to the events of interest in the input audio file, $O = g(\mathbf{x})$. In our case these events will be bat echolocation calls. Each prediction from the model, $\mathbf{o} \in O$, represents a distinct event and contains information characterising the time, frequency, and semantic (i.e. species) components of the event. Specifically, each predicted event, $\mathbf{o} = [t_{\text{start}}, t_{\text{end}}, f_{\text{min}}, f_{\text{max}}, \mathbf{p}_{\text{species}}]$, represents the start time, end time, minimum frequency, maximum frequency of the event, along with probability vector indicating which species the model thinks is present. Here, $\mathbf{p}_{\text{species}}$ is a $C+1$ dimensional vector that sums to one, and represents the probability of the species the model thinks emitted the call, for each one of C different species of interest plus one additional background class (i.e. 'Not bat'). Note, that this representation is distinct from conventional acoustic classification models that only attempt to determine the species present in a short duration input spectrogram, i.e. $y = g(\mathbf{x})$, where $y \in \{1, ..., C+1\}$ is an integer denoting the predicted species label.

[1] Code available at `https://github.com/macaodha/batdetect2`

2.2 | Detection and classification model

We implement our joint classification and detection model $g()$ as a deep neural network. Our model is inspired by computationally efficient one-stage object detection methods from computer vision, e.g. Zhou et al. (2019). Unlike two-stage methods that first propose a set of possible events of interest and then assign each event to a class (i.e. a species), one-stage approaches directly predict the location and size of each event (i.e. echolocation call) in the input.

Our model makes use of a U-Net-style architecture (Ronneberger et al., 2015), with an encoder that extracts features from the input spectrogram, followed by a decoder that generates the predicted size and location of each echolocation call along with the corresponding species' probabilities. The model also uses skip connections which facilitate the sharing of higher resolution feature information (in terms of frequency and time) from the encoder to the decoder. The output of the decoder is a distribution over time and frequency indicating where the model thinks a set of calls are present and also the sizes (in terms of frequency range and duration) of the calls. As a final step, we pass this output to a non-maximal suppression layer, implemented via max pooling, in order to extract the local peak detections (Zhou et al., 2019). This step prevents the model from predicting multiple calls very close to each other (i.e. within a few milliseconds). A high-level depiction of the model is illustrated in Figure 1.

A common issue with many current deep learning-based bat call detection and classification models, e.g. (Mac Aodha et al., 2018; Chen et al., 2020; Kobayashi et al., 2021), is that they typically only utilise very short temporal input windows (e.g. less than 50 milliseconds) to determine if a species is present. This prevents these models from reasoning about inter-pulse temporal information that can exist between individual calls and can span hundreds or thousands of milliseconds. This issue could be partially addressed by using more computationally expensive backbone encoder models that have a larger temporal receptive field size, e.g. (Simonyan and Zisserman, 2015; He et al., 2016). However, the

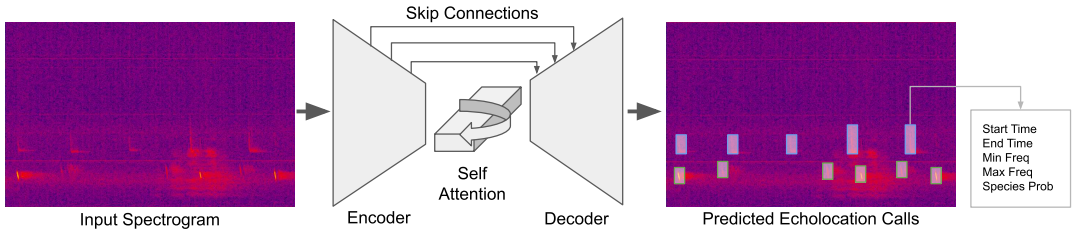

FIGURE 1 Overview of BatDetect2, our echolocation call classification and detection model. The model consists of a convolutional neural network-based encoder and decoder with skip connections that share extracted features from the encoder to decoder. It utilises a self-attention layer in the middle of the model so that it can reason over a longer temporal scale. In contrast to most existing deep learning-based bat call classifiers, our model directly predicts the time in file of each event of interest, along with the duration of the event, the frequency range, and the species.

downside of such models is that they are much larger, and thus have more parameters that need to be trained. This larger size necessitates larger supervised training datasets and results in a reduction in speed at inference time. To overcome this problem, without having to increase the size and capacity of the encoder, we introduce a self-attention layer into the middle of our network. Transformer-based self-attention architectures (Vaswani et al., 2017) are among the current most performant models in natural language processing owning to their ability to capture long-range dependencies that occur in the input data. The introduction of this layer allows our model to 'attend' to information from different points in time in the input audio file in order to increase or decrease its estimated likelihood that a given species is present at the current time step. Note that this self-attention layer only operates along the temporal dimension and is thus very computationally efficient.

Our entire model is trained end-to-end using a three component loss function which includes a detection loss, a classification loss, and an event size loss. The first two losses are implemented using a focal loss (Lin et al., 2017), and the final one uses an L1 penalty. The model and associated training and evaluation code are implemented using the `PyTorch` deep learning framework (Paszke et al., 2019). A detailed description of the audio pre-processing steps, model architecture, training losses, and training settings are provided in the supporting information.

2.3 | Audio annotation interface

Our model requires supervision in the form of bounding boxes encompassing each individual echolocation call present in an audio file. In order to obtain this, we developed an audio annotation interface to enable human annotators to efficiently draw boxes and to assign a species class label to every audible echolocation call in a given input file. The interface is implement using the `Flask` web framework (Flask, 2021) and is is depicted in Figure 2. This framework allows us to deploy the interface on the web to allow annotators to annotate remotely or we can also deploy it locally on an annotator's own device.

The interface has been optimised to speed up the annotation process. For example, we pre-cache the spectrogram generation step for the next file to be annotated so that the annotator does not have to wait when switching between files. In addition, it is possible to change the spectrogram visualisation settings in order to trade-off frequency resolution for temporal resolution, or vice versa. The annotations are stored together in a separate JSON file for each audio file using a format similar to the one used for the COCO dataset (Lin et al., 2014). Annotators can play the audio file using a time expansion factor of ten to ensure that the ultrasonic signals of interest are audible.

Unless otherwise specified, the audio files that we annotated had information at the file-level related to which species were present in the recording. Annota-

tors were instructed to draw boxes around each individual echolocation call, irrespective of how faint the call was. They then assigned the recording-level species class label to an annotation unless it differed from a prototypical echolocation call for that species. Harmonics were not annotated as part of the main call. In cases where it was not possible to assign the correct class label, or when multiple species were present in a file, annotators marked unknown calls as being from a generic 'Bat' class.

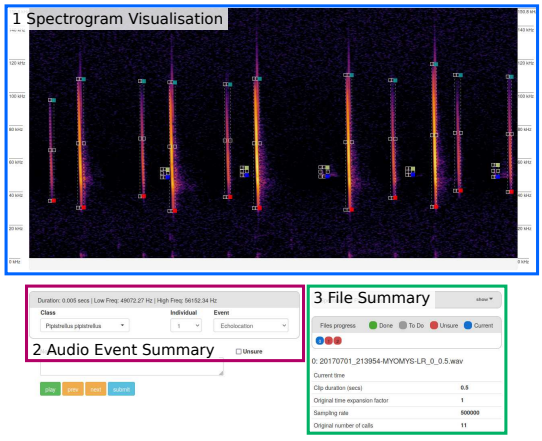

FIGURE 2 Our audio annotation interface has three main components: (i) spectrogram visualisation and playback, (ii) editing of echolocation call annotations, and (iii) file-level metadata display.

2.4 | Audio datasets

We train and evaluate our model on five different full spectrum ultrasonic acoustic datasets. Additional details for each, including visual examples and per-species counts, are available in the supporting information.

2.4.1 | UK data

This dataset contains audio data from 17 bat species that breed in the UK, and has been collated from six different sources. In total there are 2,809 distinct audio files, with an average duration of 1.04 seconds, and the dataset contains a total of 34,635 annotated echolocation calls. To increase our robustness to background noise, we supplement this data with 4,225 additional 0.384 second duration files from the iBats Program (Jones et al., 2013). This adds an additional 6,842 annotated bat calls that do not have a confirmed species label. Finally, we also add 345, one second duration, empty files (i.e. no bats present) from London, UK, collected using the recording devices described in Gallacher et al. (2021). Our model is capable of using these non-species bat recordings to learn better audio representations.

We split the UK data into two different train and test sets, UK_{same} and UK_{diff}. For UK_{same} we randomly assign files to the test set, ensuring a maximum of four files per species, per data source. The remaining files are kept for the training set. This results in 7,010 train files and 369 test files, containing 36,955 and 4,522 calls respectively. UK_{diff} is a more challenging split. Here we hold-out the largest single data source for testing. This leaves 5,911 training and 1,468 test files, containing 24,315 and 17,162 echolocation calls. This second split represents a more challenging test-case where the data is guaranteed to be very different from the training set. This also results in a reduction in the overall amount of training data, both in terms of sheer quantity but also diversity. Both variants of the dataset retain the 4,570 files without species labels as part of their respective training sets.

2.4.2 | Yucatan data

The second dataset consists of 1,193 one second audio clips extracted from 285 passive acoustic recordings from the Yucatan peninsula in Mexico. The data was collected as part of a study by MacSwiney G et al. (2008). It is smaller in size than the UK dataset, but is representative of the type of data that would be feasible to collect and annotate as part of a smaller-scale monitoring project. The annotations from the original study were used and then expanded to ensure that all audible echolocation events were annotated. The final annotated dataset contains 10,020 echolocation calls from

17 different species. We divided the data into 911 training and 282 test clips, making sure to separate at the original recording-level, and not the clip-level, to ensure that clips from the same recording were not in both sets.

2.4.3 | Australia data

This next dataset consists of a set of 14 bat species which can be found in the major cotton growing region on the north west plains of New South Wales and adjacent areas in central southern Queensland. Bat calls were recorded in the field from individuals released after capture, following positive species identification. This dataset features species with similar call characteristics which makes it particularly challenging. The data was randomly split at the file level, with 80% of the recordings for a species staying the train set, and the rest in the test. This resulted in 4,569 and 1,327 individual calls in the train and test sets respectively.

2.4.4 | Brazil data

Our final dataset represents an orthogonal challenge to the first two. It contains 320 recordings of ten second duration each collected between January and March 2019 in south-eastern Brazil using AudioMoth recorders (Hill et al., 2018). Here we have access to the recordings but do not have any confirmed species metadata. As a result, instead of annotating the calls with species labels, we instead created 'meta-categories' based on the dominant frequency component exhibited by each call. This resulted in three distinct call groups in the final annotated dataset. Like the other datasets, this annotation was performed manually, where the protocol again stipulated that all echolocation call instances in each recording should be annotated. We split the data into 256 train files and 64 test files, which resulted in 7,989 and 2,010 calls respectively.

2.5 | Baseline model

In order to evaluate the effectiveness of our model, we compare it to a traditional bat call parameter/feature extraction pipeline. To do this, we use the Tadarida-D model from Bas et al. (2017), which consists of two main components: (i) a bat echolocation call detector and (ii) a echolocation call feature extractor. The extracted call features are a set of numerical values that encode information about the shape and frequency content of each individual detected bat call. In the case of Tadarida-D, this amounts to 268 features for each detected event. For additional details about the specific set of call features in Tadarida-D, please consult the original paper (Bas et al., 2017).

For each of our datasets, we first run Tadarida-D to detect the calls and extract the call features. Then for each detected event in the training set, we compute the overlap between the event (using the reported time in file, duration, and frequency range from Tadarida-D) and our ground truth annotations. We select the detection that overlaps most in time and frequency with a given ground truth annotation and then assign the species label from the ground truth to that event. If a detected event does not match to a ground truth annotation it is assigned to the 'Not bat' class. Each ground truth annotation can only be assigned to one predicted detection. Finally, we train a Random Forest (Breiman, 2001) classifier on the extracted calls using the implementation from `scikit-learn` (Pedregosa et al., 2011) with default parameters. It is important to emphasise that while we are using Tadarida-D, our baseline is *not* directly equivalent to the full Tadarida method as we do *not* make use of their pre-trained models, labeling interface, classification code, or post-processing steps. However, this baseline allows us to control for the impact of the training data as we can ensure that we are using the same audio and ground truth annotations at training and test time for both our method and this baseline that uses Tadarida's features. This baseline also cannot make use of the additional echolocation events that only have the generic 'Bat' class label. However, this is only relevant for the UK datasets.

2.6 | Evaluation metrics

We use four different evaluation metrics to quantify the performance of our model. The first, detection average precision ('AP Det'), evaluates the ability of the model to correctly identify all valid echolocation calls in the test data. This metric calculates the precision and recall resulting from varying a threshold on the model output predictions for the 'Bat' versus 'Not bat' task. We then average over these different thresholds to quantify the area under the precision-recall curve, using the interpolation method used in Everingham et al. (2010). A prediction is counted as a true positive if its estimated start time overlaps with a ground truth echolocation call by at most ten milliseconds. This is the same evaluation criteria used in Mac Aodha et al. (2018).

'AP Det' does not evaluate the ability of the model to accurately assign the correct species label to a prediction. To address this, we also report the mean average precision across the classes ('mAP Class'). This involves taking the per-class average precision and then averaging this value over each class. This also has the added effect of weighting each class equally, irrespective of the number of calls for each class in the test set. Here, we exclude calls for which there are no ground truth species labels available.

'mAP Class' suffers from one major limitation. As the classes are evaluated independently, it does not highlight cases where the underlying model may be poorly calibrated and thus require different output thresholds for each class. Calibration issues like this can result from class-level data imbalances in the training data. To overcome this limitation, we also report a third precision based metric which we refer to as 'Top Class'. Here we simply take the top predicted class label, along with its corresponding probability, for each detected call and then evaluate the average precision as above. Unlike 'mAP Class', this metric can be biased if there is a large imbalance in the classes in the test set.

The final metric, 'File Acc', evaluates the file-level classification accuracy. For this metric only, we exclude test files that have been manually annotated as containing more than one species. In order to convert the multiple possible individual call predictions for a given file into a single file-level class label, we threshold each of the individual detections and remove any detection below the threshold. We then sum the per-class probabilities of the remaining detections and choose the class with the highest sum as the file-level prediction. Finally, we report the file-level accuracy corresponding to the single best threshold across all files. The best possible score for each of these four metrics is 1.0, and the worst is 0.0.

3 | RESULTS

3.1 | Detection and classification performance

In Table 1 we present the main results comparing the performance of our model, BatDetect2, to the Random Forest baseline that uses Tadarida-D call features. The results represent the average of three different models, each trained with different random initialisations. We observe that across all datasets, and the four evaluation metrics, BatDetect2 performs best. The Random Forest baseline also performs well on the comparatively easy Brazil dataset, but struggles on the other four. The difference in performance is between 0.05 and 0.37 mean average precision ('mAP Class'), across the datasets.

We can see that BatDetect2's detection performance, reported via 'AP Det', is strong. This indicates that the model is capable of correctly detecting the vast majority of calls. However the lower performance for the two call-level classification metrics ('mAP Class' and 'Top Class') indicates that it can have difficulty identifying the correct species for a given call in some situations. Table 1 also highlights the challenge posed by the more difficult UK_{diff} dataset in contrast to the performance on UK_{same}. In the supporting information, we illustrate the impact that the amount of training data per-class has on test performance, and broadly observe that more data increases performance. The comparatively worse performance on the Yucatan and Australia datasets can likely partially be explained by the challenging set of species contained within each and the smaller set of distinct training files available.

TABLE 1 Performance of our BatDetect2 model compared to the Random Forest baseline with uses traditional bat echolocation call features. We evaluate both models using the same five test datasets. For each of the metrics, higher numbers are better, and the results are averaged over three runs. BatDetect2 performs best in all cases.

	BatDetect2 (Ours)				Random Forest Baseline			
Dataset	**AP Det**	**mAP Class**	**Top Class**	**File Acc**	**AP Det**	**mAP Class**	**Top Class**	**File Acc**
UK_{same}	0.971	0.884	0.843	0.866	0.890	0.706	0.638	0.800
UK_{diff}	0.964	0.810	0.690	0.780	0.903	0.587	0.47	0.687
Yucatan	0.923	0.803	0.818	0.861	0.649	0.430	0.467	0.682
Australia	0.973	0.700	0.640	0.795	0.928	0.603	0.507	0.719
Brazil	0.926	0.962	0.940	1.000	0.883	0.912	0.910	1.000

In Figure 3 we display the per-class precision-recall curves for BatDetect2. We also show precision-recall curves at the genus-level. For these genus results, we do not retrain the models, but instead sum the predictions for each species belonging to a given genus, convert the ground truth class label to the genus label, and then evaluate in the same way as the species-level curves. We also display the file-level confusion matrix. As with the 'File Acc' metric, we only report results for files that have one reported species in them. By comparing the genus-level results in the second column for the challenging *Myotis* calls to the corresponding species-level ones in the first column for both UK datasets, we observe that the model is capable of resolving the classification task to the genus-level for these calls, but has difficulty for some at the species level. This difficulty is most apparent when looking at the confusion matrix for UK_{diff} in the second row of Figure 3. Here we see that our model confuses some *Myotis* species at the file-level.

We visualise the model's predictions for a subset of files in Figure 4. We observe that BatDetect2 is capable of detecting faint calls, and also handles situations where multiple species are present in a recording. The model is also robust to background noise. This is most apparent in the example from the Brazil dataset recorded using an AudioMoth (Hill et al., 2018) on the bottom row of the figure. In this example we can see a repetitive high frequency signal, most prominent at ~60kHz that repeats every 50 milliseconds. Despite this structured noise, our model does not produce any false positives in this example.

It takes BatDetect2 just under four minutes to process and save the results for 424, ten second duration, 384kHz AudioMoth recordings using a GPU, i.e. 70.6 minutes of ultrasonic data in total. Tadarida-D, which does not utilise a GPU, takes 2.5 minutes for detection and feature extraction for the same data. Note that this processing time does not include the evaluation of the Random Forest. This benchmarking was performed on a workstation which contained an Intel i7-6850K CPU and an Nvidia TITAN Xp GPU.

3.2 | Impact of self-attention

In Table 2 we present results a variant of the model on the UK_{diff} dataset where we remove the self-attention layer, i.e. 'No Self-Attn'. Again we report performance averaged over three different runs. We observe a large drop in performance when compared to the full model. Notably, the call detection results measured by 'AP Det' are not impacted, but two of the classification metrics, 'mAP Class' and 'Top Class', show a large decrease when removing this component. This points to the value of longer temporal range reasoning when resolving species identity that is provided by the self-attention layer. In the supporting information, we provide a visualisation of how the self-attention layer makes use of information from different points in time in order to improve its species-level predictions.

TABLE 2 Performance of two different variants of BatDetect2 on the UK_{diff} test set. The results in the first row are the same as the BatDetect2 results in the second row of Table 1. 'No Self-Attn' is the same as the full BatDetect2 model but where the self-attention layer has been removed at training and test time.

Dataset	AP Det	mAP Class	Top Class	File Acc
Full model	0.964	0.810	0.690	0.780
No Self-Attn	0.962	0.725	0.614	0.790

4 | DISCUSSION

4.1 | Model performance

BatDetect2 performs significantly better than the traditional call feature-based baseline tested. For the vast majority of species in the UK_{same} dataset, BatDetect2 results in high precision at high recall rates (see Figure 3). This is important as it enables practitioners to trade-off recall for precision to ensure that they obtain reliable, high confidence, predictions from the model. The file-level accuracy is 78% and 86.6% for the UK_{diff} and UK_{same} datasets, where a large percentage of the mistakes can be attributed to known challenging species, i.e. the *Myotis* species. While we observe a performance drop for UK_{diff}, the UK_{same} results indicate that training on larger quantities of more representative data results in a more effective model.

Unlike existing deep learning-based classifiers, our model produces interpretable predictions in the form of time and frequency boxes around the detected calls (see Figure 4). This is valuable as it will enable practitioners to inspect the model predictions to better understand any failure cases they may observe for their datasets. BatDetect2 can efficiently use information from longer input time scales via the self-attention layer without significantly increasing the amount of computation performed at test time. This results in a model that can perform inference ~17 times faster than real time using a GPU, i.e. 17 minutes of recorded ultrasonic audio takes one minute to fully process.

Perhaps most importantly, we showed that the same pipeline, without any modifications, can be applied to audio data from four distinct regions. This is valuable as it will allow practitioners to focus on collecting and annotating datasets for their species of interest. Our annotation interface assists this process and will enable researchers to make annotations available to others in a standardised and open format.

4.2 | Limitations

BatDetect2 performs well across the five datasets tested, however it still suffers from some limitations. We rely on the availability of diverse, and exhaustively annotated, training data. Collecting such data can be challenging, in addition to being time consuming to annotate. This limitation is common to any supervised learning-based method. While methods for semi-supervised and self-supervised training offer the potential to learn effective models with limited to no training supervision, diverse labelled data is still needed to evaluate the performance of the developed models. Bat calls can exhibit plasticity depending on the population sampled (Montauban et al., 2021), the presence of other species, and the composition of the local environment. As a result, care needs to be taken to ensure that the collected training datasets are representative of the downstream deployment situations as much as possible. Finally, our training datasets currently only contain annotated echolocation calls, and thus the model cannot make predictions for other types of calls, e.g. social calls or feeding buzzes. With appropriate training data, this could be addressed.

For a given input recording, BatDetect2 returns a list of detections along with their corresponding time in file and species probabilities. It is left up to the user to decide how to best merge the individual detections into a set of 'bat passes', where a pass constitutes a sequence of individual calls. This summary step can be important, as practitioners often derive statistics based on the number of detected individuals (which is difficult to ascertain) or their activity, as opposed to the number of detected calls. One approach is to use a grouping-based heuristic based on the time between detected calls as in Mac Aodha et al. (2018). The high recall rates of BatDe-

tect2 means that this type approach is less likely to separate individual bat passes into multiple different ones. In contrast, methods that produce high numbers of false negatives run the risk of overcounting the number of passes as they can miss faint calls in a sequence, and thus incorrectly break them up into a number of shorter passes.

5 | CONCLUSIONS

We presented BatDetect2, a general-purpose model for detecting and classifying bat echolocation calls in challenging high-frequency audio data. We showed that the same model, without modifications, can be trained and evaluated on data from different geographical regions. In addition to pre-trained models, we also make data and code for our models and annotation interface available to stimulate future research.

Acknowledgements

Thanks to all the data providers and annotators for their efforts. Thanks also to Grant Van Horn for helpful discussions.

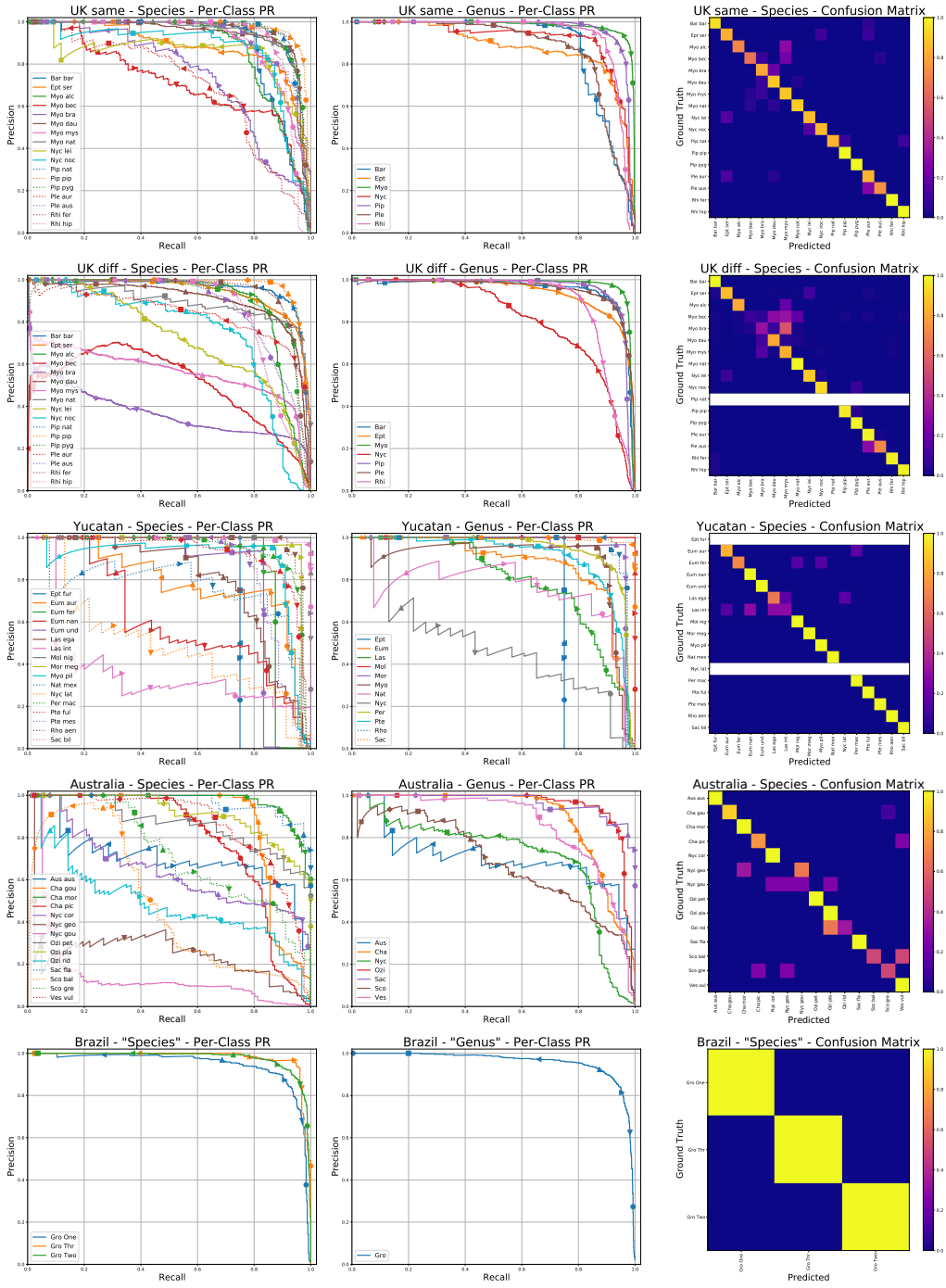

FIGURE 3 Precision-recall (PR) and confusion matrices for our BatDetect2 model for the five different test sets. The first column depicts the per-species precision-recall curves and the second column is the per-genus equivalent. The third column illustrates the file-level confusion matrix, where white rows indicate that there were no species of that type in the filtered test set. Each row depicts a different dataset.

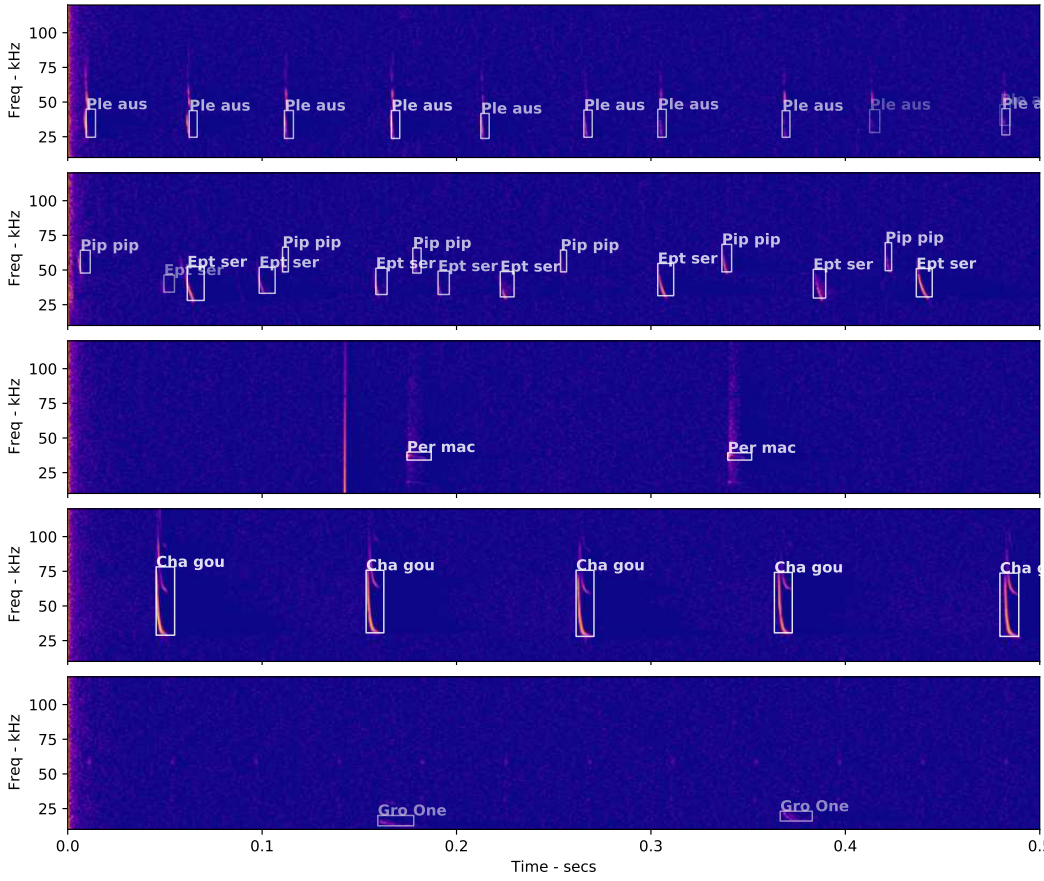

FIGURE 4 Predictions from our BatDetect2 model. Each row represents a different audio file selected from the test sets of the UK_{same}, UK_{diff}, Yucatan, Australia, and Brazil datasets, ordered from top to bottom. The intensity of an individual predicted bounding box indicates the model's confidence, with a brighter white value indicating more confident. The text above each box corresponds to the highest probability class label.

References

Aide, T. M., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G. and Alvarez, R. (2013) Real-time bioacoustics monitoring and automated species identification. *PeerJ*, **1**, e103.

Bas, Y., Bas, D. and Julien, J.-F. (2017) Tadarida: a toolbox for animal detection on acoustic recordings. *Journal of open research software*.

Breiman, L. (2001) Random forests. *Machine learning*, **45**, 5–32.

Chen, X., Zhao, J., Chen, Y.-h., Zhou, W. and Hughes, A. C. (2020) Automatic standardized processing and identification of tropical bat calls using deep learning approaches. *Biological Conservation*, **241**, 108269.

Christin, S., Hervet, É. and Lecomte, N. (2019) Applications for deep learning in ecology. *Methods in Ecology and Evolution*, **10**, 1632–1644.

Dierckx, L., Beauvois, M. and Nijssen, S. (2022) Detection and multi-label classification of bats. In *International Symposium on Intelligent Data Analysis*, 53–65.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J. and Zisserman, A. (2010) The pascal visual object classes (voc) challenge. *International journal of computer vision*.

Flask (2021) Python web framework. `https://flask.palletsprojects.com`.

Frick, W. F., Kingston, T. and Flanders, J. (2020) A review of the major threats and challenges to global bat conservation. *Annals of the New York Academy of Sciences*, **1469**, 5–25.

Gallacher, S., Wilson, D., Fairbrass, A., Turmukhambetov, D., Mac Aodha, O., Kreitmayer, S., Firman, M., Brostow, G. and Jones, K. (2021) Shazam for bats: Internet of things for continuous real-time biodiversity monitoring. *IET Smart Cities*.

Gibb, R., Browning, E., Glover-Kapfer, P. and Jones, K. E. (2019) Emerging opportunities and challenges for passive acoustics in ecological assessment and monitoring. *Methods in Ecology and Evolution*, **10**, 169–185.

He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep residual learning for image recognition. In *Conference on computer vision and pattern recognition*.

Hill, A. P., Prince, P., Piña Covarrubias, E., Doncaster, C. P., Snaddon, J. L. and Rogers, A. (2018) Audiomoth: Evaluation of a smart open acoustic device for monitoring biodiversity and the environment. *Methods in Ecology and Evolution*, **9**, 1199–1211.

Ioffe, S. and Szegedy, C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*.

Jones, G., Jacobs, D. S., Kunz, T. H., Willig, M. R. and Racey, P. A. (2009) Carpe noctem: the importance of bats as bioindicators. *Endangered species research*, **8**, 93–115.

Jones, G. and Siemers, B. M. (2011) The communicative potential of bat echolocation pulses. *Journal of Comparative Physiology A*, **197**, 447–457.

Jones, K. E., Russ, J. A., Bashta, A.-T., Bilhari, Z., Catto, C., Csősz, I., Gorbachev, A., Győrfi, P., Hughes, A., Ivashkiv, I., Koryagina, N., Kurali, A., Langton, S., Collen, A., Margiean, G., Pandourski, I., Parsons, S., Prokofev, I., Szodoray-Paradi, A., Szodoray-Paradi, F., Tilova, E., Walters, C. L., Weatherill, A. and Zavarzin, O. (2013) *Indicator Bats Program: A System for the Global Acoustic Monitoring of Bats*, chap. 10, 211–247.

Khalighifar, A., Gotthold, B. S., Adams, E., Barnett, J., Beard, L. O., Britzke, E. R., Burger, P. A., Chase, K., Cordes, Z., Cryan, P. M. et al. (2022) Nabat ml: Utilizing deep learning to enable crowdsourced development of automated, scalable solutions for documenting north american bat populations. *Journal of Applied Ecology*.

Kingma, D. P. and Ba, J. (2015) Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kobayashi, K., Masuda, K., Haga, C., Matsui, T., Fukui, D. and Machimura, T. (2021) Development of a species identification system of japanese bats from echolocation calls using convolutional neural networks. *Ecological Informatics*, **62**, 101253.

Law, H. and Deng, J. (2018) Cornernet: Detecting objects as paired keypoints. In *European conference on computer vision*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017) Focal loss for dense object detection. In *International conference on computer vision*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L. (2014) Microsoft

coco: Common objects in context. In *European conference on computer vision*.

Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A. and Yosinski, J. (2018) An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*.

Lostanlen, V., Salamon, J., Cartwright, M., McFee, B., Farnsworth, A., Kelling, S. and Bello, J. P. (2018) Per-channel energy normalization: Why and how. *Signal Processing Letters*, **26**, 39–43.

Mac Aodha, O., Gibb, R., Barlow, K. E., Browning, E., Firman, M., Freeman, R., Harder, B., Kinsey, L., Mead, G. R., Newson, S. E., Pandourski, I., Parsons, S., Russ, J., Szodoray-Paradi, A., Szodoray-Paradi, F., Tilova, E., Girolami, M., Brostow, G. and Jones, K. E. (2018) Bat detective—deep learning tools for bat acoustic signal detection. *PLOS Computational Biology*, **14**, 1–19.

MacSwiney G, M. C., Clarke, F. M. and Racey, P. A. (2008) What you see is not what you get: the role of ultrasonic detectors in increasing inventory completeness in neotropical bat assemblages. *Journal of applied Ecology*, **45**, 1364–1371.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E. and Nieto, O. (2015) librosa: Audio and music signal analysis in python. In *Python in science conference*, vol. 8, 18–25.

Montauban, C., Mas, M., Tuneu-Corral, C., Wangensteen, O. S., Budinski, I., Martí-Carreras, J., Flaquer, C., Puig-Montserrat, X. and López-Baucells, A. (2021) Bat echolocation plasticity in allopatry: a call for caution in acoustic identification of pipistrellus sp. *Behavioral Ecology and Sociobiology*, **75**, 1–15.

Nair, V. and Hinton, G. E. (2010) Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*.

Obrist, M. K. and Boesch, R. (2018) Batscope manages acoustic recordings, analyses calls, and classifies bat species automatically. *Canadian Journal of Zoology*, **96**, 939–954.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D. and Le, Q. V. (2019) Specaugment: A simple augmentation method for automatic speech recognition. In *Interspeech*.

Parsons, S. and Jones, G. (2000) Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. *Journal of experimental biology*, **203**, 2641–2656.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019) Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, **32**, 8026–8037.

Paumen, Y., Mälzer, M., Alipek, S., Moll, J., Lüdtke, B. and Schauer-Weisshahn, H. (2021) Development and test of a bat calls detection and classification method based on convolutional neural networks. *Bioacoustics*, 1–12.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Prat, Y., Taub, M. and Yovel, Y. (2016) Everyday bat vocalizations contain information about emitter, addressee, context, and behavior. *Scientific Reports*, **6**, 1–10.

Roemer, C., Julien, J.-F. and Bas, Y. (2021) An automatic classifier of bat sonotypes around the world. *Methods in Ecology and Evolution*, **12**, 2432–2444.

Ronneberger, O., Fischer, P. and Brox, T. (2015) U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241.

Russo, D., Ancillotto, L. and Jones, G. (2018) Bats are still not birds in the digital era: echolocation call variation and why it matters for bat species identification. *Canadian Journal of Zoology*, **96**, 63–78.

Russo, D., Salinas-Ramos, V. B., Cistrone, L., Smeraldo, S., Bosso, L. and Ancillotto, L. (2021) Do we need to use bats as bioindicators? *Biology*, **10**, 693.

Sechidis, K., Tsoumakas, G. and Vlahavas, I. (2011) On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

Simonyan, K. and Zisserman, A. (2015) Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Stowell, D. (2022) Computational bioacoustics with deep learning: a review and roadmap. *PeerJ*, **10**.

Tabak, M. A., Murray, K. L., Reed, A. M., Lombardi, J. A. and Bay, K. J. (2021) Automated classification of bat echolocation call recordings with artificial intelligence. *Ecological Informatics*, 101526.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017) Attention is all you need. In *Advances in neural information processing systems*.

Walters, C. L., Collen, A., Lucas, T., Mroz, K., Sayer, C. A. and Jones, K. E. (2013) Challenges of using bioacoustics to globally monitor bats. In *Bat evolution, ecology, and conservation*, 479–499.

Walters, C. L., Freeman, R., Collen, A., Dietz, C., Brock Fenton, M., Jones, G., Obrist, M. K., Puechmaille, S. J., Sattler, T., Siemers, B. M. et al. (2012) A continental-scale tool for acoustic identification of european bats. *Journal of Applied Ecology*, **49**, 1064–1074.

Wang, Y., Getreuer, P., Hughes, T., Lyon, R. F. and Saurous, R. A. (2017) Trainable frontend for robust and far-field keyword spotting. In *International Conference on Acoustics, Speech and Signal Processing*.

Zamora-Gutierrez, V., Balvanera, S. M., Esquivelzeta, E. R. et al. (2021) The evolution of acoustic methods for the study of bats. In *50 Years of Bat Research*, 43–59.

Zamora-Gutierrez, V., Lopez-Gonzalez, C., MacSwiney Gonzalez, M. C., Fenton, B., Jones, G., Kalko, E. K., Puechmaille, S. J., Stathopoulos, V. and Jones, K. E. (2016) Acoustic identification of mexican bats based on taxonomic and ecological constraints on call design. *Methods in Ecology and Evolution*, **7**, 1082–1091.

Zhang, H., Cisse, M., Dauphin, Y. N. and Lopez-Paz, D. (2018) mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Zhou, X., Wang, D. and Krähenbühl, P. (2019) Objects as points. *arXiv preprint arXiv:1904.07850*.

Zualkernan, I., Judas, J., Mahbub, T., Bhagwagar, A. and Chand, P. (2020) A tiny cnn architecture for identifying bat species from echolocation calls. In *International Conference on Artificial Intelligence for Good*, 81–86.

— (2021) An aiot system for bat species classification. In *International Conference on Internet of Things and Intelligence System*, 155–160.

Supporting Information

A | IMPLEMENTATION DETAILS

A.1 | Model architecture

The full architecture of our BatDetect2 model is outlined in Table S1. The input to the model is a two dimensional spectrogram, and the output is a prediction for each frequency and time bin in the spectrogram indicating which species of bat the model predicts is echolocating there, if any, and the predicted frequency range and time duration of the detected echolocation event. The model is fully convolutional, so can operate on arbitrary length sequences, but in practice we chunk longer input audio files into clips that are less than two second long, and then process each clip independently.

The model uses a U-Net (Ronneberger et al., 2015) style architecture, with an encoder and decoder with skip connections between then. In the middle bottleneck of the model there is a self-attention layer (Vaswani et al., 2017), `self_attn`, that enables the model to share information across longer time scales. This is particularly valuable for bat species classification as other calls in the sequence can be a useful signal in aiding the classification of a given species. Most current deep learning based models for bat call detection only use very narrow input time windows (e.g. only 23 milliseconds in the case of Mac Aodha et al. (2018)) and are thus unable to capture these longer range temporal dependencies. The self-attention layer uses a feature dimension of 256 and does not include any additional positional encoding information.

Our model makes use of two building block layers, `CoordConvDown` and `CoordConvUp`. At a high level, the `CoordConvDown` layer takes a tensor as input and returns a spatially downsized version of it as an output. Conventional convolutions are translation invariant which is not necessarily a desirable property when attempting to determine the species of bat that is present in an audio recording. This is because the absolute frequency of the event is a valuable discriminative signal. To overcome this issue, the `CoordConvDown` layer appends non-learnable coordinates to the vertical (i.e. frequency) dimension. Note that unlike the original CoordConv paper (Liu et al., 2018), here we do not add coordinates to the temporal dimension as we want the model to be invariant to the time that the events of interest occur in the recording.

The specific operations performed in a `CoordConvDown` layer are as follows: append frequency coordinate information, 2D convolution, 2×2 max pool downsampling, batch normalisation (BN) (Ioffe and Szegedy, 2015), followed by ReLU (Nair and Hinton, 2010) non-linearity.

The `CoordConvUp` layer performs similar operations but in reverse, i.e. it upsamples the input tensor. The specific operations are: 2D bilinear upsampling, appending frequency coordinates, 2D convolution, batch normalisation, followed by ReLU.

The model returns two outputs, the predicted class $\hat{Y}$ and the estimated temporal duration and frequency range of the event $\hat{S}$. The predicted class vector includes an additional background class (i.e. 'Not bat') and the vector sums to one for each time and frequency bin. We run a simple non-maximal suppression operation on the output so that we only report the local maximum for each detected event, i.e. we wish to suppress multiple predictions that are nearby in time and frequency and only select one. This is achieved by running a two dimensional max pooling operation with a kernel size of 9×9. The model then reports the top 200 events, ordered by detection probability, for each one second of input audio.

TABLE S1 Description of the full architecture for our BatDetect2 model. The values for input and output size refer to the feature dimension, height, and width of the respective tensors (e.g. (1, 128, 512) is one feature channel, with height 128 and width 512). The kernel size is represented as height and width. In the case where two tensors are added together for the input to a layer, this is simply performed using an element wise addition. The model outputs a $C + 1$ dimensional vector for each location in time and frequency, where $C + 1$ represents the number of classes plus one additional class for background, i.e. 'Not bat'. The model also outputs an additional two dimensional vector for each location which encodes the predicted width (i.e. duration) and height (i.e. frequency range) of any echolocation event at that location in time and frequency.

layer name	input	layer type	input size	output size	kernel size
Encoder					
conv_down_0	spectrogram	CoordConvDown	(1, 128, 512)	(32, 64, 256)	(3,3)
conv_down_1	conv_down_0	CoordConvDown	(32, 64, 256)	(64, 32, 128)	(3,3)
conv_down_2	conv_down_1	CoordConvDown	(64, 32, 128)	(128, 16, 64)	(3,3)
Bottleneck					
conv_3	conv_down_2	Conv2d, BN, ReLU	(128, 16, 64)	(256, 16, 64)	(3,3)
conv_1d	conv_3	Conv2d, BN, ReLU	(256, 16, 64)	(256, 1, 64)	(16,1)
self_attn	conv_1d	Self-Attention	(256, 1, 64)	(256, 1, 64)	n/a
repeat_vert	self_attn	Repeat Vertical	(256, 1, 64)	(256, 16, 64)	n/a
Decoder					
conv_up_0	repeat_vert + conv_3	CoordConvUp	(256, 16, 64)	(64, 32, 128)	(2,2)
conv_up_1	conv_up_0 + conv_down_1	CoordConvUp	(64, 32, 128)	(32, 64, 256)	(2,2)
conv_up_2	conv_up_1 + conv_down_0	CoordConvUp	(32, 64, 256)	(32, 128, 512)	(2,2)
Output					
conv_op_0	conv_up_2	Conv2d, BN, ReLU	(32, 128, 512)	(32, 128, 512)	(3,3)
pred_class - $\hat{Y}$	conv_op_0	Conv2d, Softmax	(32, 128, 512)	($C + 1$, 128, 512)	(1,1)
pred_size - $\hat{S}$	conv_op_0	Conv2d, ReLU	(32, 128, 512)	(2, 128, 512)	(1,1)

A.2 | Audio pre-processing

Here we outline the steps we perform in order to convert the raw input audio samples into the spectrogram before it is input into the model. After loading from disk, the input audio is resampled to 256kHz using the 'polyphase' method from `librosa` (McFee et al., 2015). We then compute the magnitude spectrogram using a short-time Fourier transform with a window size of 512 samples (assuming the audio has been resampled to 256kHz) and use a window overlap of 75%. The bat echolocation calls of interest only occur within a specific range of frequency bands. As a result, we only retain the bands between 10kHz and 120kHz. To provide robustness with respect to volume changes we normalise the spectrogram using Per-Channel Energy Normalisation (PCEN) (Wang et al., 2017). Lostanlen et al. (2018) showed this to be more effective than traditional logarithmic-based normalisation. Similar to Aide et al. (2013) and Mac Aodha et al. (2018), we also subtract the mean value from each frequency band to remove the impact of any constant background noise. As a final step, we use bilinear interpolation to resize the temporal dimension down

by a factor of two and resample the frequency bands into 128 bins. In the end, an input audio file of one second in duration results in a spectrogram of size 128×1024.

A.3 | Training loss

In this section we describe the training loss used by our model. The loss function is composed of three main terms and is inspired by those used in the CenterNet method for object detection in images (Zhou et al., 2019). The combined losses encourages the model to correctly predict the location, in frequency and time, of each echolocation call, the duration and frequency range of the call, and the species that is responsible for making the call.

Let us denote $\mathbf{x} \in R^{H \times W}$ as our input spectrogram, with height H and width W. Here, height refers to the number of frequency bins and width is the number of temporal bins in the spectrogram. Prior to the final post-processing step (i.e. non-maximal suppression), our model outputs two tensors, $\hat{Y} \in [0,1]^{H \times W \times C+1}$ and $\hat{S} \in \mathbb{R}_{\geq 0}^{H \times W \times 2}$. Here, C is the total number of species of interest, and we add an additional class to represent the background class (i.e. no bat present). $\hat{Y}$ is the predicted species class probabilities and $\hat{S}$ contains the predicted size of any echolocation call estimated to be present. At training time we have access to the ground truth values for Y and S which we use to train the model. Both $\hat{Y}$ and $\hat{S}$ contain an estimated value for each location in time and frequency space in the input spectrogram. For example, for a given frequency band f and time step t, $\hat{S}_{ft1}$ encodes the predicted duration of the call (i.e. $t_{\text{end}} - t_{\text{start}}$), and $\hat{S}_{ft2}$ encodes the predicted frequency range of the call (i.e. $f_{\text{max}} - f_{\text{min}}$). For a description of how Y and S are generated, please see Section A.4.

We also define $\hat{E}_{ft} = \sum_{c=1}^{C} \hat{Y}_{ftc}$, and similarly $E_{ft} = \sum_{c=1}^{C} Y_{ftc}$. $\hat{E}$ and E represent predicted and ground truth class-agnostic echolocation call scores, i.e. 'Bat' versus 'Not bat'. Note that for $\hat{E}$ and E, we only sum over the classes one to C, and do not include the background class. We include these additional terms as there are many instances in which our annotators have difficulty determining the correct species id for a given call, and thus they can only label the event with the generic 'Bat' class label. We can still make use of this supervision by allowing the model to determine which species may be present. Our goal during training is to minimise the difference between our estimated $\hat{E}$, $\hat{Y}$, and $\hat{S}$ and the respective ground truth values E, Y, and S. If successful, the model will be able to correctly predict the location in time and frequency of any echolocation call along with the species id of the bat that generated the call.

A.3.1 | Losses

Our first loss encourages the model to correctly discriminate between bat echolocation calls and non-bat calls, i.e. background noise or other vocalising species. To achieve this, we use the focal loss (Lin et al., 2017). Specifically, we use the keypoint variant of the focal loss from Law and Deng (2018), which is defined as

$$L_{det} = -\frac{1}{N} \sum_{f=1}^{H} \sum_{t=1}^{W} \begin{cases} (1 - \hat{E}_{ft})^{\alpha} \log(\hat{E}_{ft}) & \text{if } E_{ft} = 1 \\ (1 - E_{ft})^{\beta} (\hat{E}_{ft})^{\alpha} log(1 - \hat{E}_{ft}) & \text{otherwise,} \end{cases} \tag{1}$$

where N is the number of echolocation events in the spectrogram.

Our next loss penalises the model for assigning the wrong species label to a detected echolocation call. This loss is similar L_{det}, but instead of only discriminating between 'Bat' and 'Not bat', this loss encourages the model to predict the correct species label for each echolocation call. We use a masked version of the loss which is only applied to

locations in the spectrogram where there is a echolocation call present, i.e. where $E_{ft} > 0$.

$$L_{class} = -\frac{1}{N}\sum_{f=1}^{H}\sum_{t=1}^{W}\sum_{c=1}^{C+1}\begin{cases}0 & \text{if } E_{ft} = 0\\ (1-\hat{Y}_{ftc})^{\alpha}\log(\hat{Y}_{ftc}) & \text{if } E_{ft} > 0 \text{ and } Y_{ftc} = 1\\ (1-Y_{ftc})^{\beta}(\hat{Y}_{ftc})^{\alpha} log(1-\hat{Y}_{ftc}) & \text{otherwise.}\end{cases} \tag{2}$$

The final component of our loss penalises the model for incorrectly predicting the 'size' of the predicted bounding box which overlaps with a ground truth echolocation call. Like L_{class}, this loss is only applied to locations in time and frequency where we have observed an echolocation call in the training set.

$$L_{size} = \frac{1}{N}\sum_{f=1}^{H}\sum_{t=1}^{W}\begin{cases}\left|\hat{S}_{ft1} - S_{ft1}\right| + \left|\hat{S}_{ft2} - S_{ft2}\right| & \text{if } \sum_k S_{ftk} > 0\\ 0 & \text{otherwise.}\end{cases} \tag{3}$$

Here, $\sum_k S_{ftk} > 0$ simply indicates that this size loss is only applied where there is a echolocation call present.

The final combined loss we aim to minimise during training is

$$L = \lambda_1 L_{det} + \lambda_2 L_{class} + \lambda_3 L_{size}. \tag{4}$$

We sum this loss over each spectrogram in a given input training batch. During training, we set λ_1, λ_2, λ_3 to 1.0, 2.0, and 0.1 respectively, and for both focal losses we set $\alpha = 2$ and $\beta = 4$.

A.4 | Training details

The model and training code are implemented in `PyTorch` (Paszke et al., 2019). We train our model end-to-end using the Adam optimizer (Kingma and Ba, 2015), starting with an initial learning rate of 0.001 and use a cosine annealing learning rate schedule, a batch size of 8, and train for 200 epochs (unless specified otherwise). We perform a series of augmentations at training time to increase the variation in the input audio. These augmentations include: random linear combination of two input audio files (Zhang et al., 2018), simulated echo, random volume scaling, temporal stretching, and time and frequency masking (Park et al., 2019). The probability that any one augmentation is applied is 0.2, and multiple augmentations can be applied together. Like Zhou et al. (2019), we generate ground truth 'heatmaps' for Y, and thus E, with Gaussian kernels using a standard deviation of 2.0 for each annotated echolocation event. These heatmaps represent the regression targets Y that our model uses during training. Unlike Zhou et al. (2019), which parameterises bounding boxes using their center point, we instead use the bottom left of each echolocation call as it tends to be more stable for many species, i.e. it is typically much easier for human annotators to identify the bottom left of a call as opposed to its center. However, there are notable exceptions to this assumption, e.g. the two *Rhinolophus* species found in the UK. Finally, S contains the height and width (converted to pixel units with respect to the spectrogram size) of each call computed from their bottom left coordinates. If no call is present for a given time-frequency bin, we simply store a zero for both height and width for that location.

B | AUDIO DATASETS

Here we provide additional details of the different datasets used in our evaluation.

B.1 | UK data

B.1.1 | Audio recording

In total there are 17 species in the UK dataset. This is the total number of species which are known to be breeding in the UK. The data comes from 2,809 audio files, and contains a total of 34,635 annotated echolocation calls. The data has been collected using a variety of devices and was provided by a number of different sources. There are six main sources of data, where each source constitutes a single organisation or individual that provided multiple different audio files. This diversity is important as it maximises the variation in the training set, with the ultimate aim of having better generalisation performance at test time. The vast majority of the recordings were made in the UK, but there were also some additional files included from the species of interest that were recorded elsewhere (e.g. Europe). During the annotation phase, we prioritised annotating only one clip from each of the original input recordings, at most two seconds in duration, as opposed to densely annotating long multi-second audio files. This was also performed in order to increase the data diversity, as there can often be a large amount of self-similarity within the same longer recording. As a result, the clipped files vary in duration from between 0.4 to two seconds, and the average duration is just over one second.

In order to increase our robustness to background noise, we also supplement the UK species audio by including additional recordings that are either empty (i.e. did not contain bats) or where we only knew if a bat was present, but not which species. The empty recordings were collected in London, UK, using the custom built IoT smart sensor from Gallacher et al. (2021). In total there are 345 three second files in this set. The second set of extra data came from the iBats Program (Jones et al., 2013) as was adapted from Mac Aodha et al. (2018). This set includes 4,225 files of 0.384 seconds in duration and contains 6,842 annotated bat calls. This data was recorded using Tranquility Transect detector using a time expansion factor of ten.

B.1.2 | Annotation

The audio files were annotated using our browser-based annotation tool described in the main paper. With the exception of the background and bat-only recordings, the rest of the files contained confirmed species at the file level. Experienced annotators, familiar with the characteristics of UK bat echolocation calls, drew boxes around each individual echolocation call. When unsure of the species label, they annotated the call using the generic 'Bat' class label.

The BatDetect2 model predicts the location of the lower left corner for each echolocation call in an input recording. For the two constant call frequency-based species in the UK, *Rhinolophus ferrumequinum* and *Rhinolophus hipposideros*, there was a high degree of variability in the position of the lower left corner of the call. This happens as a direct result of the recording quality, characteristics of the local environment, and the distance of the bat from the microphone As a result, it was often very difficult to determine the exact lower frequency for these two species. To overcome this issue, we standardised the lower and upper frequency for each of the these species by setting them to per-species mean values, where the means were computed on the training sets.

B.1.3 | Data split

We constructed two splits for the UK dataset. Both splits contain the same number of calls overall, and only differ in how the data is distributed between their respective training and test sets. As noted earlier, there are six main sources of data for our UK bat recordings. The first split, referred to as UK_{same}, simply shuffles the files randomly into training and test sets and ensures that there is a maximum of four recordings (i.e. files not calls) per species, per data source, in the test set. This results in a split with 7,010 training files and 369 test files.

The second split, UK_{diff}, is more challenging. Here we simulate a difficult real world setting where an entire data source is held out for validation. We remove one of our largest sources, which leaves 5,911 training files and 1,468 test files. This increases the difficulty due to the reduction in the training set size as well as increasing any potential domain gap that may exists between the train and test sets. This test set does not contain one of the species, *Pipistrellus nathusii*, as it was not possible to capture any recordings of it. Note that in both cases the data is still split at the file level (as opposed to individual call level). This minimises any potential overlap between the training and test sets.

A summary of the number of calls per species can be found in Tables S2 and S3. Figures S4 depicts a per-class average spectrogram for each species in the training set for the UK_{diff} split. Note that this averaging hides many of the recording specific difficulties and noise. It is thus is only provided for illustrative purposes as it shows the dominant 'shape' of the call for each species.

B.2 | Yucatan data

B.2.1 | Audio recording

This dataset consists of 285 passive recordings gathered in the Yucatan peninsula in Mexico as part of a field study conducted between 2004 and 2006 (MacSwiney G et al., 2008). A Pettersson D980 bat detector device was used to detect and record bat calls. The device was active throughout three ten-minute periods at night, in a total of eight sites and covering twelve sampling nights per site. When active, and if a bat call was detected, the device would record for three seconds and a time expanded version would be stored on a magnetic tape. The recordings were then cut into one second clips, resulting in a total of 1,193 audio files.

B.2.2 | Annotation

The species identification of the bat calls was made in two phases. For the original study, all recordings were reviewed manually. From each recording, at most five representative echolocation calls per detected species was selected and analyzed using Bat Sound Pro 3.10. The species of each call was then identified through comparison to a bat call library of captured bats from the same study. Please consult MacSwiney G et al. (2008) to see the full details of their identification protocol.

In the second phase we annotated all missing bat calls using our annotation interface. Bounding boxes were drawn around each detected bat call in the spectrogram. Species identification was performed by comparing to the previously annotated calls. In order to gain confidence on the species labels for the additional boxes, we evaluated the identification accuracy of the human annotators. A species label was added only if the human annotator could accurately identify said species (precision above 95%). In cases where it was not possible to determine the species, the call was labelled using the generic 'Bat' class. A recording was fully annotated when all bat echolocation calls were marked with a bounding box and all recognisable calls were tagged with its species, or the generic, label. This resulted in a total of 1,193 audio clips that were fully annotated and kept as part of the dataset. Three species (Pteronotus

TABLE S2 Number of annotated echolocation calls in the UK dataset using the UK_{same} split. There are a total of 7,010 and 369 training and test files, each containing 36,955 and 4,522 annotated echolocation calls respectively.

id	species name	num train calls	num test calls
0	Bat	8112	203
1	Barbastellus barbastellus	864	179
2	Eptesicus serotinus	2374	211
3	Myotis alcathoe	695	183
4	Myotis bechsteinii	648	222
5	Myotis brandtii	1775	166
6	Myotis daubentonii	5729	640
7	Myotis mystacinus	2430	384
8	Myotis nattereri	2384	328
9	Nyctalus leisleri	1056	85
10	Nyctalus noctula	310	99
11	Pipistrellus nathusii	1224	236
12	Pipistrellus pipistrellus	1653	245
13	Pipistrellus pygmaeus	2171	396
14	Plecotus auritus	917	193
15	Plecotus austriacus	690	177
16	Rhinolophus ferrumequinum	1915	290
17	Rhinolophus hipposideros	2008	285

personatus, Molossops greenhalli, and Molossus sinaloae) were excluded as they only appeared in fewer than seven distinct recordings. The annotations for these species was set to the generic 'Bat' class. The final annotated dataset consists of 10,020 individual bat echolocation calls with bounding box annotations from 17 different species.

B.2.3 | Data split

To train and evaluate the detection and classification models we split the dataset into distinct training and testing subsets. To minimize any leakage from the test to the train set, we opted to split the data at the recording level, i.e. we avoided including one-second clips from the same recording in the training and testing subsets. The test set contains ~20% (282 audio clips) of all recordings while the remaining ~80% (911 audio clips) was used for training. In order to maintain the distribution of calls per species between the full dataset and the testing and training datasets, we labeled each recording with all its occurring species and used a stratified sampling method for multilabel datasets (Sechidis et al., 2011). A summary of the number of calls per species can be found in Table S4, and Figure S5 provides a visual summary for each species.

TABLE S3 Number of annotated echolocation calls in the UK dataset using the UK_{diff} split. There are a total of 5,911 and 1,468 training and test files, each containing 24,315 and 17,162 annotated echolocation calls respectively.

id	species name	num train calls	num test calls
0	Bat	7501	814
1	Barbastellus barbastellus	468	575
2	Eptesicus serotinus	403	2182
3	Myotis alcathoe	374	504
4	Myotis bechsteinii	241	629
5	Myotis brandtii	351	1590
6	Myotis daubentonii	3998	2371
7	Myotis mystacinus	1378	1436
8	Myotis nattereri	2610	102
9	Nyctalus leisleri	695	446
10	Nyctalus noctula	209	200
11	Pipistrellus nathusii	1460	0
12	Pipistrellus pipistrellus	868	1030
13	Pipistrellus pygmaeus	1461	1106
14	Plecotus auritus	528	582
15	Plecotus austriacus	331	536
16	Rhinolophus ferrumequinum	717	1488
17	Rhinolophus hipposideros	722	1571

B.3 | Australia data

B.3.1 | Audio recording

The Australian dataset used to train and test the model was taken from a bat call reference library collected by one of the co-authors. The subset used consists of a set of 14 bat species which have a sympatric distribution in the major cotton growing region on the north west plains of New South Wales and adjacent areas in central southern Queensland. Bat calls were recorded in the field from individuals released after capture, following positive species identification. A custom made digital ultrasound recorder from Nanobat Systems was used to record echolocation calls in 5 second sequences with a sampling rate 500 kHz and stored as 16 bit WAVs. Bats were recorded for as long as they flew around the release site until out of recording range. The resulting files were analysed and edited using Audacity 3.2.0 to find echolocation pulse sequences with good signal to noise ratio, undistorted waveforms and as close to search phase as possible. Edited wav files were then accumulated from the release recordings of multiple individuals of the same species and across the species group.

TABLE S4 Number of annotated echolocation calls in the Yucatan dataset. In total there are 911 and 282 training and test files, which contain 7,677 and 2,343 individual calls respectively.

id	species name	num train calls	num test calls
0	Bat	3556	1236
1	Eptesicus furinalis	94	4
2	Eumops auripendulus	156	36
3	Eumops ferox	60	24
4	Eumops nanus	66	33
5	Eumops underwoodi	36	18
6	Lasiurus ega	250	69
7	Lasiurus intermedius	106	31
8	Molossus nigricans	65	25
9	Mormoops megalophylla	172	30
10	Myotis pilosatibialis	519	90
11	Natalus mexicanus	62	26
12	Nyctinomops laticaudatus	98	23
13	Peropteryx macrotis	1036	322
14	Pteronotus fulvus	509	167
15	Pteronotus mesoamericanus	345	81
16	Rhogeessa aeneus	166	36
17	Saccopteryx bilineata	381	92

B.3.2 | Annotation

All of the edited length audio sequence files for the entire dataset were annotated using the browser-based annotation tool described in the main paper. These audio files had an average length of 3.29 seconds, with the shortest being 0.23 seconds and the longest being 10 seconds in duration. All annotated pulses were labelled by species since the original sequences were obtained from individually released bats, identified to species level. The only exception comes from the Ozimops species where the low release number of individuals (rarely caught) was augmented by identifying species from additional field recordings of bat activity at night. This was done manually by conventional sound analysis of field recordings taken from various study areas and using an experienced bat bioacoustics expert familiar with this genus. There were some instances where multiple species may have been present in a given file, and thus were potentially incorrectly attributed to the wrong species label.

B.3.3 | Data split

The data was randomly split at the file level, with 80% of the recordings for a species staying the train set, and the rest in the test. This resulted in 220 training and 60 testing files. A summary of the number of calls per species can be found in Table S5, and Figure S6 illustrates a visual summary for each species.

TABLE S5 Number of annotated echolocation calls in the Australia dataset. In total there are 220 and 60 training and test files, which contain 4,569 and 1,327 individual calls respectively.

id	species name	num train calls	num test calls
0	Bat	180	18
1	Austronomus australis	125	35
2	Chalinolobus gouldii	568	146
3	Chalinolobus morio	429	155
4	Chalinolobus picatus	327	157
5	Nyctophilus corbeni	537	101
6	Nyctophilus geoffroyi	179	41
7	Nyctophilus gouldi	363	97
8	Ozimops petersi	149	42
9	Ozimops planiceps	142	52
10	Ozimops ridei	122	64
11	Saccolaimus flaviventris	131	40
12	Scotorepens balstoni	232	120
13	Scotorepens greyii	273	90
14	Vespadelus vulturnus	812	169

B.4 | Brazil data

B.4.1 | Audio recording

Data for this study was collected between January and March 2019 in south-eastern Brazil. The data used for training is a subset of acoustic data collected using AudioMoth (Hill et al., 2018) recorders which were set to record at a sampling rate of 395 kHz for one minute every five minutes between 22:00 and 04:00. The recorders were deployed on coffee plantations and in adjacent forest fragments. The final dataset consists of 320 ten second audio recordings.

B.4.2 | Annotation

The echolocation calls for this dataset were again annotated used our annotation interface. As no species labels were available for this dataset, we opted to group the calls based on their dominant frequency. Specifically, calls

TABLE S6 Number of annotated echolocation calls in the Brazil dataset. In total there are 256 files in the training set and 64 in the test set. In both cases the files are ten seconds in duration.

id	species name	num train calls	num test calls
0	Bat	1646	619
1	Group One	2168	490
2	Group Two	2993	742
3	Group Three	1182	159

were initially labelled to genus level where quality allowed, but were later merged to a coarser call type groups. This resulted in three distinct groups, along with the generic bat class which served as an additional class for cases where it was not possible to identify calls to one of the previously mentioned three groups.

B.4.3 | Data split

We randomly assigned ~80% of the audio files (256 files) to the training set and the remaining ~20% (64 files) to the test set. This resulted in a total of 7,989 and 2,010 calls in the respective sets. A summary of the number of calls per group can be found in Table S6, and Figure S7 illustrates a visual summary per call group.

C | ADDITIONAL RESULTS

C.1 | Visualising self-attention

In Figure S1 we illustrate the self-attention mechanism is action for one file. The attention module only operates along the temporal dimension. For each point in time it computes the self-attention scores with all other time steps. In this example we can see that the attention reveals a strong affinity with similar calls at other time points in the input. The model can thus make use of this global information when estimating which species is present locally.

C.2 | Impact of amount of training data

In Figures S2 and S3 we plot the test-time per-species average precision against the number of calls for each species seen at training time. Figure S2 depicts the UK_{same} data split, and we broadly observe an increase in performance as we increase the number of training calls. In the case of UK_{diff}, we also see stronger performance with more training calls. However, here it is also worth noting that there are some poor performing *Myotis* species that are challenging to classify.

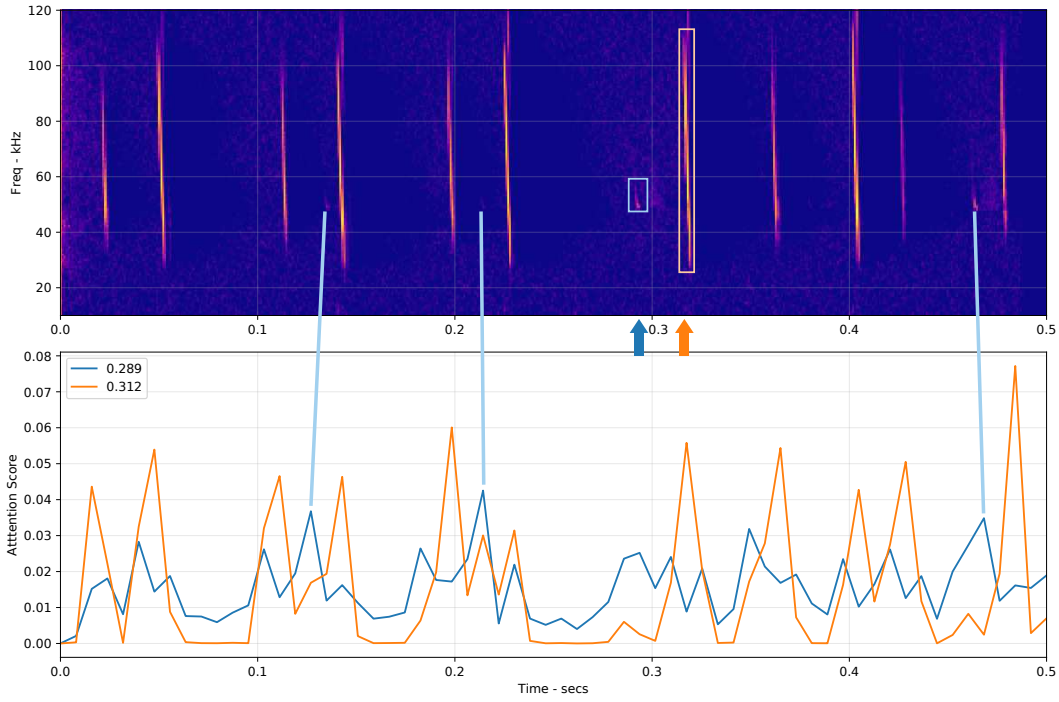

FIGURE S1 Visualisation of the self-attention scores for one audio file from the UK dataset. Here we show the attention weights for only two locations in the input - at 0.289 and 0.312 seconds in the input spectrogram. We denote these two time points with a blue and orange arrow respectively, along with showing bounding boxes on the calls. The attention scores corresponding to the two time points are illustrated with blue and orange lines on the bottom plot. The orange line shows high attention for the other *Myotis* calls and the blue line indicates that the model places more attention on the other, less prominent, *Pipistrelle* calls. Note, there appears to be a very faint *Pipistrelle* call before the 0.4 second time step that the model has a low attention score for.

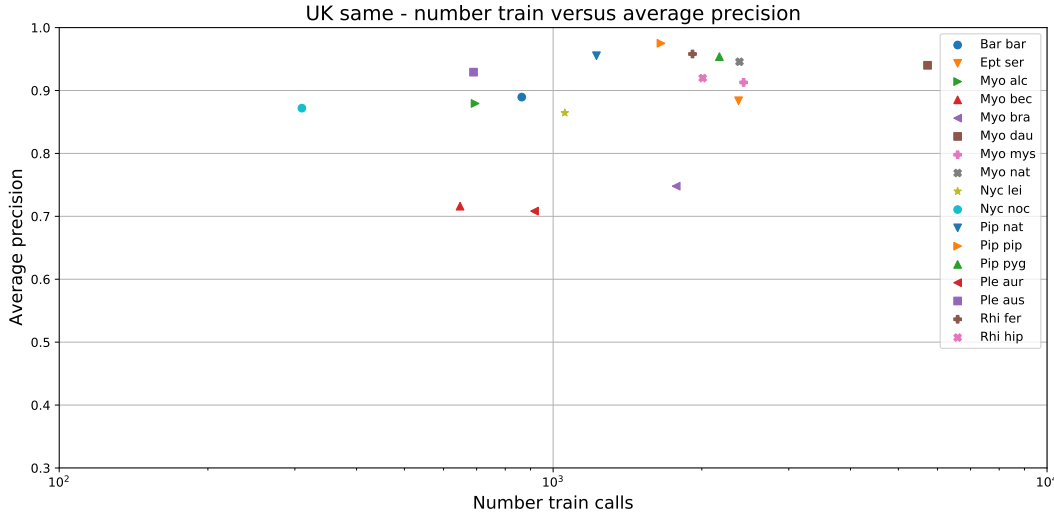

FIGURE S2 Test-time per-species average precision versus the number of training calls for the UK_{same} data split. Note that the horizontal axis is log scaled.

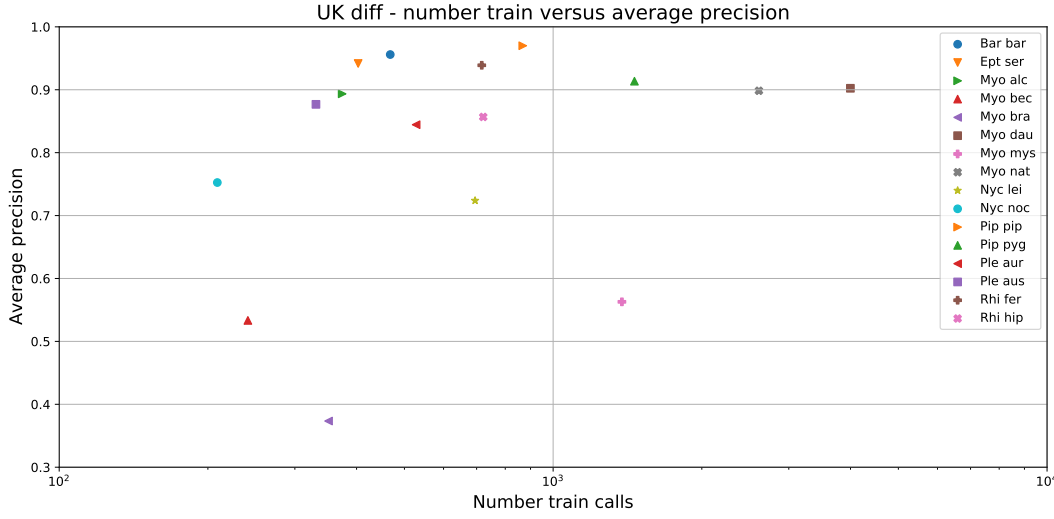

FIGURE S3 Test-time per-species average precision versus the number of training calls for the UK_{diff} data split. Note that the horizontal axis is log scaled.

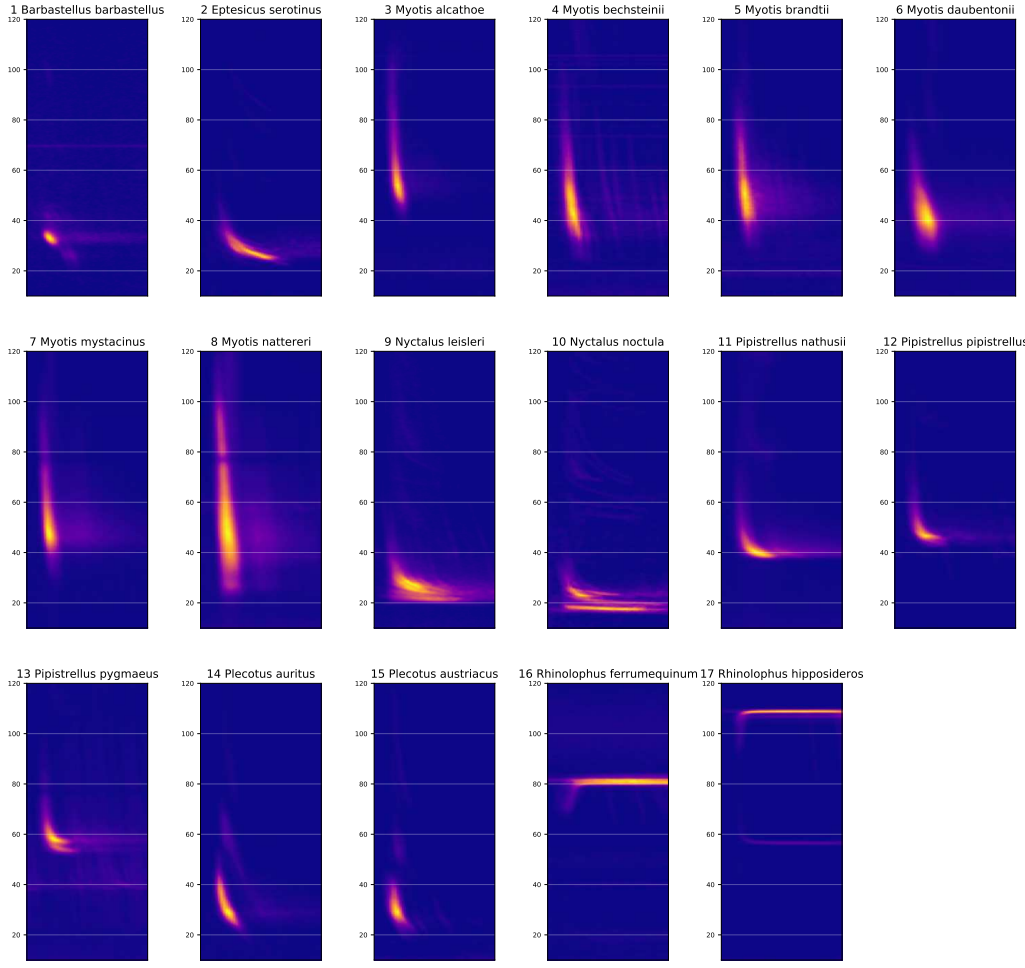

FIGURE S4 Visualisation of the UK_{diff} species split. Here, each sub-image represents the average spectrogram for each echolocation call from that species in the training set. The vertical axis represents kHz, and spans 10kHz to 120kHz, and the time duration for each spectrogram is 33.5 milliseconds.

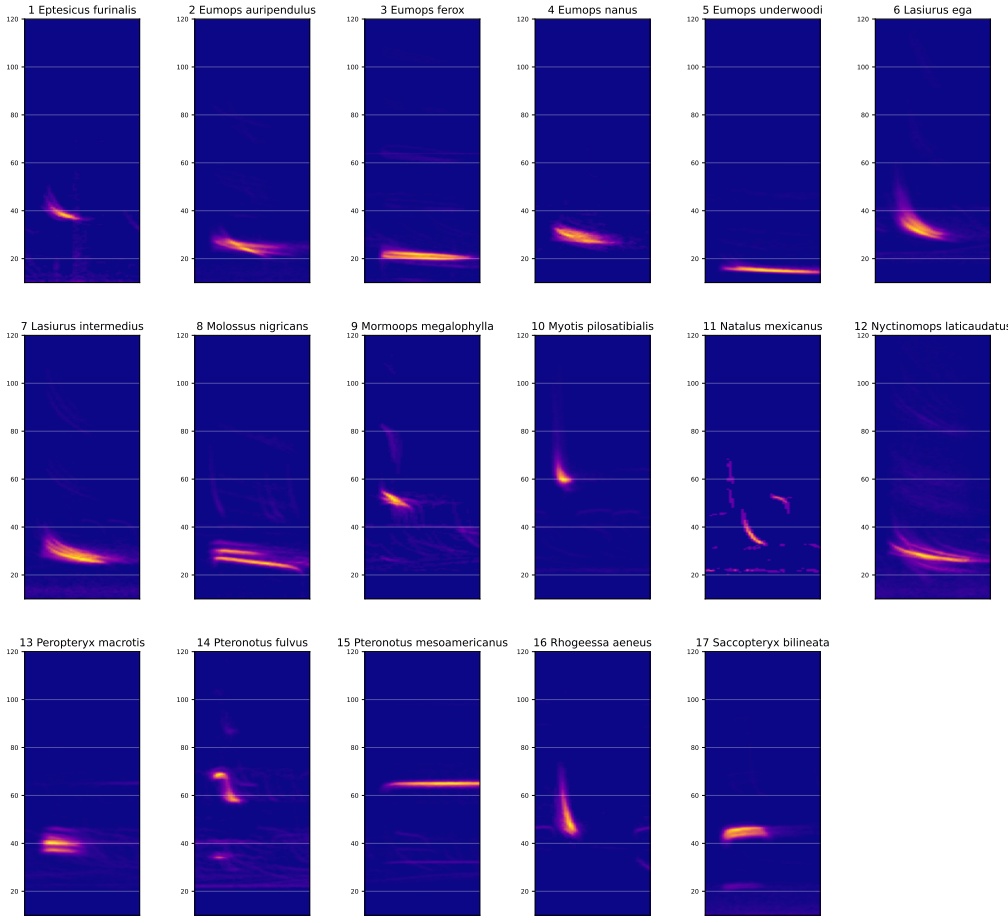

FIGURE S5 Visualisation of Yucatan species. Here, each sub-image represents the average spectrogram for each echolocation call from that species in the training set. The vertical axis represents kHz, and spans 10kHz to 120kHz, and the time duration for each spectrogram is 33.5 milliseconds. Note that for some species we have limited numbers of example calls which results in noisy average spectrograms.

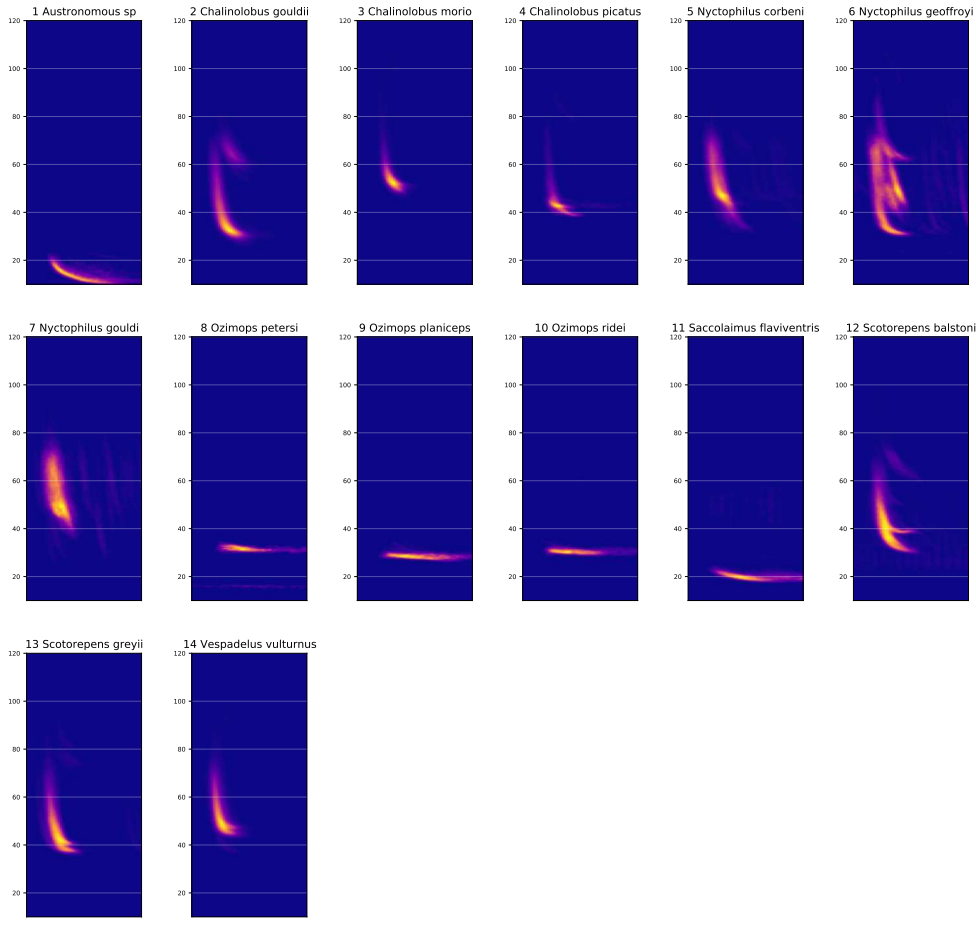

FIGURE S6 Visualisation of Australian species. Here, each sub-image represents the average spectrogram for each echolocation call from that species in the training set. The vertical axis represents kHz, and spans 10kHz to 120kHz, and the time duration for each spectrogram is 33.5 milliseconds.

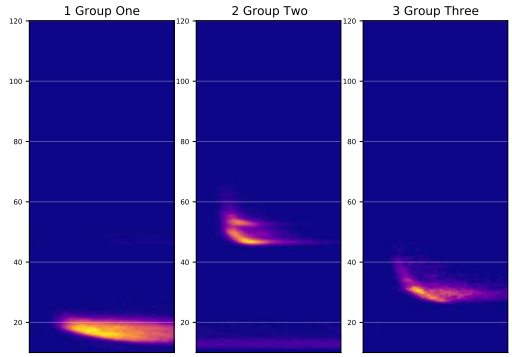

FIGURE S7 Visualisation of the Brazil data. Here, the spectrograms do not represent species, but instead three distinct groups of calls. Each sub-image represents the average spectrogram for each echolocation call from that group in the training set. The vertical axis represents kHz, and spans 10kHz to 120kHz, and the time duration for each spectrogram is 33.5 milliseconds.